Tight Bounds for Online TSP on the Line

Antje Bjelde, Yann Disser, Jan Hackfeld, Christoph Hansknecht, Maarten Lipmann, Julie Meissner, Kevin Schewior, Miriam Schlöter, Leen Stougie

Tight Bounds for Online TSP on the Line

Antje Bjelde [*†] Yann Disser[‡] Jan Hackfeld[*§] Christoph Hansknecht[¶]
Maarten Lipmann[‖] Julie Meißner[* **] Kevin Schewior[†† ‡‡] Miriam Schlöter[*§]
Leen Stougie[§§]

Abstract

We consider the online traveling salesperson problem (TSP), where requests appear online over time on the real line and need to be visited by a server initially located at the origin. We distinguish between closed and open online TSP, depending on whether the server eventually needs to return to the origin or not. While online TSP on the line is a very natural online problem that was introduced more than two decades ago, no tight competitive analysis was known to date. We settle this problem by providing tight bounds on the competitive ratios for both the closed and the open variant of the problem. In particular, for closed online TSP, we provide a 1.64-competitive algorithm, thus matching a known lower bound. For open online TSP, we give a new upper bound as well as a matching lower bound that establish the remarkable competitive ratio of 2.04.

Additionally, we consider the online Dial-A-Ride problem on the line, where each request needs to be transported to a specified destination. We provide an improved non-preemptive lower bound of 1.75 for this setting, as well as an improved preemptive algorithm with competitive ratio 2.41.

Finally, we generalize known and give new complexity results for the underlying offline problems. In particular, we give an algorithm with running time $\mathcal{O}(n^2)$ for closed offline TSP on the line with release dates and show that both variants of offline Dial-A-Ride on the line are NP-hard for any capacity $c \geq 2$ of the server.

1 Introduction

In the online Traveling Salesperson Problem (TSP) on the line, we consider a server initially located at the origin of the real line that has to serve requests that appear over time. The server has unit speed and serves requests (in any order) by moving to the position of the corresponding request at some time after its release. The objective in online TSP on the line is to minimize the makespan, i.e., the time until all requests have been served. In the *closed* variant of the problem, the server needs to return to the origin after serving all requests, while the *open* variant has no such requirement.

Online TSP is a natural online problem similar to the classical k-server problem [20]. In the latter, the order in which requests need to be served is prescribed, and the problem thus becomes trivial on the line for $k = 1$ server. In contrast, online TSP on the line is a non-trivial problem that arises in 1-dimensional collection/delivery problems. Examples include robotic welding/screwing/depositing material, horizontal/vertical item delivery systems, and the collection of objects from mass storage shelves. The online Dial-A-Ride problem additionally allows transportation requests that specify a source and destination that need to be visited by the server in this order. If the capacity of the server is finite, it limits the number of requests that can be transported simultaneously. The online Dial-A-Ride problem on the line arises, e.g., when controlling industrial or personal elevators.

While both online TSP and online Dial-A-Ride on the line are among the most natural online problems and have been studied extensively over the last two decades [2, 4, 5, 6, 7, 10, 14, 15, 16, 17], no satisfactory (tight) analysis was known for either problem in terms of competitive ratios. We address

[*]TU Berlin, Institute of Mathematics, Germany.
[†]Supported by Einstein Foundation Berlin in the framework of Matheon.
[‡]TU Darmstadt, Institute of Mathematics, Germany.
[§]Supported by DFG Priority Programme 1736 Algorithms for Big Data.
[¶]TU Braunschweig, Institute for Mathematical Optimization, Germany.
[‖]Amsterdam, Netherlands.
[**]Supported by Einstein Foundation Berlin in the framework of Matheon and by the German Science Foundation (DFG) under contract ME 3825/1.
[††]Universidad de Chile, Santiago, Chile.
[‡‡]Partially supported by the Millennium Nucleus Information and Coordination in Networks ICM/FIC RC130003.
[§§]Vrije Universiteit, Department of Econometrics and Operations Research & CWI, Amsterdam, Netherlands.

this shortcoming for TSP on the line by providing a tight upper bound for the closed variant, as well as tight bounds for the open variant. We emphasize that our results for the open and closed variant of the problem are independent and require substantially different approaches. Aside from our results for online TSP, we narrow the gaps for online Dial-A-Ride on the line by giving improved bounds. In addition to online results, we study the computational complexity of the underlying offline problems.

1.1 Our results

We have the following results[1] (cf. Tables 1 and 2) :

Tight bounds for online TSP on the line. Our main results are best-possible online algorithms for both the open and closed variant of online TSP on the line, as well as a new (tight) lower bound for the open variant. Our algorithm for the closed variant has a competitive ratio of $(9+\sqrt{17})/8 \approx 1.64$, matching a lower bound of Ausiello et al. [6] and improving on their 1.75-competitive algorithm. For open TSP on the line, we give a lower bound of 2.04 on the competitive ratio, which is the first bound strictly greater than 2. We also provide an optimal online algorithm matching this bound and improving on the 2.33-competitive algorithm by Ausiello et al. [6]. Our results settle online TSP on the line from the perspective of competitive analysis.

Improved bounds for online Dial-A-Ride on the line. Our lower bounds for online TSP on the line immediately apply to preemptive and non-preemptive online Dial-A-Ride on the line. In particular, our lower bound of 2.04 is the first bound greater than 2 for the open variant of the problem. Additionally, we provide a simple preemptive 2.41-competitive algorithm, which improves a (non-preemptive) 3.41-competitive algorithm by Krumke [15]. For the closed Dial-A-Ride variant, the lower bound of 1.64 by Ausiello et al. [6] was improved for one server with unit capacity without preemption to 1.71 by Ascheuer et al. [2]. We improve this bound further to 1.75 for any finite capacity $c \geq 1$. The best known algorithm for closed Dial-A-Ride on the line for finite capacity $c \geq 1$ is 2-competitive and was given by Ascheuer et al. [2].

New offline complexity results. Regarding offline TSP on the line with release times, Psaraftis et al. [21] showed a dynamic program that solves the open variant in quadratic time. We refute their claim that all optimal closed tours have a very simple structure with a counterexample, and we adapt their algorithm to find an optimal closed tour in quadratic time. For the non-preemptive offline Dial-A-Ride problem on the line, results have previously been obtained for the closed variant without release times. For capacity $c = 1$ Gilmore and Gomory [12] and Atallah and Kosaraju [3] gave polynomial time algorithms, and Guan [13] proved hardness for the case $c = 2$. We show that both the open and closed variant of the problem are NP-hard for *any* capacity $c \geq 2$. Additionally, we show that the case with release times and any $c \geq 1$ is NP-hard. The complexity of offline Dial-A-Ride on the line with unbounded capacity remains open.

1.2 Further related work

For the online TSP problem in general metric spaces, Ausiello et al. [6] show a lower bound of 2 on the competitive ratio for the open version and a 1.64 lower bound for the closed version, both bounds being achieved on the real line. For the open online TSP, they present a 2.5-competitive algorithm, and for the closed version they give a 2-competitive algorithm. Jaillet and Wagner [14] give 2-competitive algorithms for the closed version that can additionally deal with precedence constraints or multiple servers. Blom et al. [7] consider the closed online TSP problem on the non-negative part of the real line and present a best possible algorithm with competitive ratio 1.5. They also study a "fair" setting where the optimum does not travel outside the convex hull of the known requests, and they derive an algorithm for the real half-line with a better competitive ratio of 1.28 for this setting. Krumke et al. [17] show that there cannot be a competitive algorithm for open online TSP with the objective of minimizing the maximum flow time instead of minimizing the makespan. For the real line they define a fair setting and give a competitive algorithm for it.

The *online repairperson problem* is the open online TSP problem with the objective of minimizing the weighted sum of completion times. Feuerstein and Stougie [10] show a lower bound of 5.83 on the best-possible competitive ratio for this problem and provide a 9-competitive algorithm for the real line. Krumke et al. [16] give a best-possible online algorithm with competitive ratio 5.83 for general metric spaces.

For the the closed online Dial-A-Ride problem without preemption, Feuerstein and Stougie [10] show a lower bound of 2 for the competitive ratio in general, and present an algorithm with a best-possible competitive ratio of 2 for the case that the server has infinite capacity. Ascheuer et al. [2] analyze

[1] Parts of our results were already claimed in [19], but mostly with weaker bounds and without a conclusive proof. Nevertheless, some of our ideas are inspired by the approaches described in [19].

Table 1: Overview of our results for online TSP on the line and online Dial-A-Ride on the line.

Online	closed		open	
	lower bound	upper bound	lower bound	upper bound
online TSP on the line				
new	1.64 [5, 6]	**1.64** (Th. 3.1)	**2.04** (Th. 4.1)	**2.04** (Th. 5.1)
old		1.75 [5, 6]	2 [4, 6]	2.33 [4, 6]
Dial-A-Ride on the line				
preemptive	1.64 [5, 6]	2 [2]	**2.04** (Th. 4.1)	**2.41** (Th. 6.1)
non-preemptive	**1.75** (Th. 6.2)			3.41 [15]

Table 2: Overview of our results for offline TSP and Dial-A-Ride on the line with release times.

Offline	closed	open
TSP on the line	$\mathcal{O}(n^2)$ (Th. 7.2)	$\mathcal{O}(n^2)$ [21]
Dial-A-Ride on the line		
non-preemptive	NP-hard (Th. 7.3)	NP-hard (Th. 7.3)

different algorithms for the same setting and present a 2-competitive algorithm for any finite capacity $c \geq 1$. For minimizing the sum of completion times instead of the makespan, Feuerstein and Stougie [10] further show a lower bound of 3 for a server with unit capacity and a lower bound of 2.41 independent of the capacity. Moreover, they provide a 15-competitive algorithm for the real line and unlimited capacity. For the same objective function, Krumke et al. [16] present an algorithm with a competitive ratio of 5.83 for a server with unit capacity in an arbitrary metric space.

The offline version of the TSP problem is a well-studied NP-hard problem (e.g., see [18]). Afrati et al. [1] show that the offline traveling repairperson problem is NP-hard in general, but can be solved in time $\mathcal{O}(n^2)$ for the real line and unweighted sum of completion times objective. There are many offline variants of the Dial-A-Ride problem, differing in capacities, the underlying metric space, release times and deadlines, open versus closed tours, and in whether preemption is allowed (e.g., see [9]). The special case without release times and unit capacity is known as the *stacker crane problem*. Attalah and Kosaraju [3] present a polynomial algorithm for the closed, non-preemptive stacker crane problem on the real line. Frederickson and Guan [11] show that this problem is NP-complete on trees. Guan [13] shows that the Dial-A-Ride problem remains easy on the line with capacities larger than one if preemption is allowed, and that it remains hard on trees. Finally, Charikar and Raghavachari [8] give a $\mathcal{O}(\sqrt{c}\log n \log\log n)$-approximation for the closed Dial-A-Ride problem in metric spaces with n points and without preemption. In the same paper they claim a 2-approximation for the problem on the line, however this result seems to be incorrect (personal communication).

2 Problem definition and notation

We consider a server that moves along the real line with (at most) unit speed. We let p_t denote the position of the server at time $t \geq 0$ and assume (without loss of generality) that $p_0 = 0$. With this notation, the speed limitation of the server can equivalently be expressed via $|p_t - p_{t'}| \leq |t - t'|$ for all $t, t' \geq 0$. A series of requests $\sigma_1, \ldots, \sigma_n$ arrives over time with $\sigma_i = (a_i, b_i; t_i)$, where $t_i \geq 0$ denotes the release time of the request and $a_i, b_i \in \mathbb{R}$ denote its source and target position, respectively. For TSP, we have $a_i = b_i$ and write $\sigma_i = (a_i; t_i)$. If not stated otherwise, we assume $t_1 \leq t_2 \leq \ldots \leq t_n$. Moreover, we assume without loss of generality that $t_i \geq |a_i|$ holds because the server can not reach σ_i before time $|a_i|$ and it only helps the algorithm to know a request earlier. We further use the notation $A^{\mathrm{R}} := \max_{i=1,\ldots,n}\{a_i, b_i, 0\}$ to denote the rightmost point that needs to be visited by the server, and similarly $A^{\mathrm{L}} := \min_{i=1,\ldots,n}\{a_i, b_i, 0\}$. Here and throughout we refer to the negative direction of the real line as *left* and the positive direction as *right*.

In both TSP and Dial-A-Ride on the line, all requests need to be *served*. For TSP, we consider a request served if $p_t = a_i$ for some time $t \geq t_i$. For Dial-A-Ride, the server may collect request σ_i at time $t \geq t_i$ if $p_t = a_i$. In the preemptive Dial-A-Ride problem, the server can drop off any request

it is carrying at its current location at any time. If request σ_i is dropped off at point p at time t, we consider it to be modified to the new request $(p, b_i; t)$. In the non-preemptive Dial-A-Ride problem, the server may only drop off a request at its target location. We consider a request served if it is ever dropped off at its target location.

In TSP on the line, the behavior of the server in our algorithms at time t will mostly depend on so-called *extreme requests*. For $t \geq 0$, we denote by $\sigma^{\mathrm{R}}(t) = (a^{\mathrm{R}}(t); t^{\mathrm{R}}(t))$ the unserved request that is rightmost of the position of the server p_t, provided such a request exists, i.e., the unserved request $\sigma = (a; t')$ with $t' \leq t$, $a > p_t$, and maximizing a. Analogously, $\sigma^{\mathrm{L}}(t) = (a^{\mathrm{L}}(t); t^{\mathrm{L}}(t))$ denotes unserved request that is leftmost of the position of the server p_t. If there is more than one right-most (left-most) request, we choose the one with the largest release time.

If the server has finite capacity $c \geq 1$, it can carry at most c requests at any time. We assume that no time is needed for picking up and dropping off requests, so that the server can pick up and drop off any number of requests at the same time, as long as its capacity is not exceeded.

We refer to a valid trajectory of the server together with the description of when it picks up and drops requests as a tour T. If the tour ends at $p_0 = 0$, we call it *closed*, otherwise it is *open*. We denote the makespan of the tour T by $|T|$. The objective in the open (closed) version of both TSP and Dial-A-Ride is to find an open (closed) tour T that serves all requests and minimizes $|T|$.

In the *offline* setting, we assume all requests to be known from the start. We let T^{Opt} denote an optimal offline tour. In the *online* setting, we assume that request σ_i is revealed at its release time t_i, at which point the tour of the server until time t_i must already have been fixed irrevocably. Additionally, we assume that the total number n of requests is unknown. We measure the quality of an online algorithm via its competitive ratio, i.e., the maximum over all sequences of requests of the ratio between the makespan of the tour it produces and $|T^{\mathrm{Opt}}|$.

In order to describe the trajectory of the server, we use the notation "move(a)" for the tour that moves the server from its current position with unit speed to the point $a \in \mathbb{R}$ and the notation "waituntil(s)" for the tour that keeps the server stationary until time s. We use the operator $\oplus$ to concatenate tours. For example, if T_0 is a tour of the server that ends at time t_0 at position p_{t_0}, then $T_0 \oplus \mathrm{move}(a)$ describes the tour that ends at time $t_0 + |a - p_{t_0}|$, is identical to the tour T_0 until time t_0 and satisfies $p_t = p_{t_0} + (a - p_{t_0})(t - t_0)$ for $t_0 \leq t \leq t_0 + |a - p_{t_0}|$. Similarly, $T_0 \oplus \mathrm{waituntil}(s)$ is the tour that ends at time $\max\{t_0, s\}$, is identical to the tour T_0 until time t_0 and that satisfies $p_{t_0} = p_t$ for all $s \in [t_0, s]$. For TSP on the line, we do not explicitly specify when a request is served, but we assume that the server serves a request whenever possible, i.e., whenever the server passes the location of a request that is already released and not yet served.

We skip the proofs of our results as they are very technical. They can be found in the full version of the paper.

3 Algorithm for closed online TSP

In this section we consider the closed online TSP problem and describe a best-possible algorithm with competitive ratio $\rho = (9 + \sqrt{17})/8 \approx 1.64$, where ρ is the nonnegative root of the polynomial $4x^2 - 9x + 4$.

We start by developing some intuition for our algorithm. In the following T^{Alg} is the tour derived by an algorithm Alg. Observe that the decision of how to move the server at time t only depends on its position p_t and the location of the left- and rightmost extreme requests $\sigma^{\mathrm{L}}(t) = (a^{\mathrm{L}}; t^{\mathrm{L}}(t))$ and $\sigma^{\mathrm{R}}(t) = (a^{\mathrm{R}}; t^{\mathrm{R}}(t))$: All other requests can be served during any tour serving $\sigma^{\mathrm{L}}(t)$ and $\sigma^{\mathrm{R}}(t)$. We will show that in this setting we can assume that $a^{\mathrm{L}}(t) \leq 0$ and $a^{\mathrm{R}}(t) \geq 0$, provided these extremes exist. If $\sigma^{\mathrm{R}}(t)$ (resp. $\sigma^{\mathrm{L}}(t)$) does not exist we set $a^{\mathrm{R}}(t) = t^{\mathrm{R}}(t) = 0$ (resp. $a^{\mathrm{L}}(t) = t^{\mathrm{L}}(t) = 0$). Thus, in contrast to the initial definition of extreme requests, in our setting a leftmost extreme is always left and a rightmost extreme is always right of the origin. When both extremes exists, we have, on a high level, three possible courses of action at time t. Either we immediately decide to serve $\sigma^{\mathrm{L}}(t)$ and $\sigma^{\mathrm{R}}(t)$ in one of the two possible orders, or we wait for some time for additional information to make a more informed decision. Intuitively, the critical case for our competitiveness is the case where we decide to serve $\sigma^{\mathrm{L}}(t)$ and $\sigma^{\mathrm{R}}(t)$ in a different order than T^{Opt}. Let $T_{\mathrm{RL}}(t)$ and $T_{\mathrm{LR}}(t)$ be the tours that start at the origin at time 0 and then move as follows,

$$\begin{aligned} T_{\mathrm{RL}}(t) = {} & \mathrm{waituntil}(t^{\mathrm{R}}(t) - |a^{\mathrm{R}}(t)|) \oplus \mathrm{move}(a^{\mathrm{R}}(t)) \\ & \oplus \mathrm{move}(a^{\mathrm{L}}(t)) \oplus \mathrm{move}(p_0), \\ T_{\mathrm{LR}}(t) = {} & \mathrm{waituntil}(t^{\mathrm{L}}(t) - |a^{\mathrm{L}}(t)|) \oplus \mathrm{move}(a^{\mathrm{L}}(t)) \\ & \oplus \mathrm{move}(a^{\mathrm{R}}(t)) \oplus \mathrm{move}(p_0). \end{aligned}$$

Note that $|T_{\mathrm{RL}}(t)|$ (resp. $|T_{\mathrm{LR}}(t)|$) is a lower bound for the length of the shortest tour serving $\sigma^{\mathrm{R}}(t)$ before $\sigma^{\mathrm{L}}(t)$ (resp. $\sigma^{\mathrm{L}}(t)$ before $\sigma^{\mathrm{R}}(t)$). Say that, at time t, both extremes exist and we greedily decide to immediately start serving the extremes in the same order as $T_{\mathrm{LR}}(t)$. To see how this can fail, assume that T^{Opt} initially follows the

tour $T_{\mathrm{RL}}(t)$, but continues to move to the left after serving $\sigma^{\mathrm{L}}(t)$. The time when T^{OPT} reaches $a^{\mathrm{L}}(t)$ is $t' = t^{\mathrm{R}}(t) + |a^{\mathrm{R}}(t)| + |a^{\mathrm{L}}(t)|$, since $t^{\mathrm{R}}(t) \geq |a^{\mathrm{R}}(t)|$ by assumption. Let t_0 be the time when we reach the origin p_0 after serving $\sigma^{\mathrm{L}}(t)$, and assume that $t' \leq t_0$. Now a new request $\sigma' = (p'; t_0)$ may arrive at time t_0 and position $p' = -|a^{\mathrm{L}}(t)| - (t_0 - t') = -t_0 + t^{\mathrm{R}}(t) + |a^{\mathrm{R}}(t)|$, that the optimum can serve immediately at time t_0. We then have

$$\begin{aligned} |T^{\mathrm{OPT}}| &= t_0 + |p'| = t_0 + |a^{\mathrm{L}}(t)| + (t_0 - t') \\ &= 2t_0 - t^{\mathrm{R}}(t) - |a^{\mathrm{R}}(t)|. \end{aligned}$$

Our algorithm still needs to serve σ' and σ^{R} at time t_0, and hence

$$\begin{aligned} |T^{\mathrm{ALG}}| &= t_0 + 2|p'| + 2|a^{\mathrm{R}}(t)| \\ &= 3t_0 - 2t^{\mathrm{R}}(t). \end{aligned}$$

For the algorithm to be ρ-competitive, we need $|T^{\mathrm{ALG}}|/|T^{\mathrm{OPT}}| \leq \rho$, and we thus obtain a condition on the earliest time t_0 we may return to the origin.

FACT 3.1. *If at time t a ρ-competitive algorithm serves the leftmost extreme $\sigma^{\mathrm{L}}(t)$ first and t_0 denotes the first time the server returns to the origin after having served $\sigma^{\mathrm{L}}(t)$, then*

$$t_0 \geq t_0^L(t) := \frac{\rho|a^{\mathrm{R}}(t)| - (2-\rho)t^{\mathrm{R}}(t)}{2\rho - 3}. \tag{3.1}$$

A symmetric statement with $t_0^R(t)$ holds if the algorithm serves $\sigma^{\mathrm{R}}(t)$ first.

Fact 3.1 illustrates that waiting is sometimes necessary in order to be competitive. On the other hand, we can obviously not afford to wait too long. To quantify this, we introduce a lower bound on the length of T^{OPT}.

DEFINITION 3.1. *If both extreme request exist at time t we define the* greedy tour $T^{\mathrm{greedy}}(t)$ *at time t as*

$$T^{\mathrm{greedy}}(t) := \begin{cases} T_{\mathrm{LR}}(t), & \text{if } |T_{\mathrm{LR}}(t)| \leq |T_{\mathrm{RL}}(t)|, \\ T_{\mathrm{RL}}(t), & \text{else.} \end{cases}$$

If only $\sigma^{\mathrm{L}}(t)$ exists, we set $T^{\mathrm{greedy}}(t) := T_{\mathrm{LR}}(t)$ and we set $T^{\mathrm{greedy}}(t) := T_{\mathrm{RL}}(t)$ if only $\sigma^{\mathrm{R}}(t)$ exists.

OBSERVATION 3.1. *We have*

$$\begin{aligned} |T_{\mathrm{RL}}(t)| &= t^{\mathrm{R}}(t) + |a^{\mathrm{R}}(t)| + 2|a^{\mathrm{L}}(t)|, \\ |T_{\mathrm{LR}}(t)| &= t^{\mathrm{L}}(t) + |a^{\mathrm{L}}(t)| + 2|a^{\mathrm{R}}(t)| \end{aligned}$$

and $|T^{\mathrm{greedy}}(t)| = \min\{|T_{\mathrm{LR}}(t)|, |T_{\mathrm{RL}}(t)|\} \leq |T^{\mathrm{OPT}}|$ if both extremes exist at time t.

Assume we are still waiting at the origin at time t, i.e. $p_t = 0$. From Observation 3.1, we conclude that if $t \leq \rho|T^{\mathrm{greedy}}(t)| - 2|a^{\mathrm{R}}(t)| - 2|a^{\mathrm{L}}(t)|$, we can wait until time $\rho|T^{\mathrm{greedy}}(t)| - 2|a^{\mathrm{R}}(t)| - 2|a^{\mathrm{L}}(t)|$, and then still serve $\sigma^{\mathrm{R}}(t)$ and $\sigma^{\mathrm{L}}(t)$ and return to the origin p_0 until time $\rho|T^{\mathrm{greedy}}| \leq \rho|T^{\mathrm{OPT}}|$, i.e., we can stay ρ-competitive. Formally, we make the following definition.

DEFINITION 3.2. *Let the* safe tour $T^{\mathrm{safe}}(t)$ *at time t be defined as*

$$T^{\mathrm{safe}}(t) := \begin{cases} T^{\mathrm{wait}} \oplus T_{\mathrm{LR}}(t), & \text{if } |a^{\mathrm{L}}(t)| \geq |a^{\mathrm{R}}(t)|, \\ T^{\mathrm{wait}} \oplus T_{\mathrm{RL}}(t), & \text{else,} \end{cases}$$

with

$$T^{\mathrm{wait}} := \mathrm{waituntil}(\rho|T^{\mathrm{greedy}}(t)| - 2|a^{\mathrm{R}}(t)| - 2|a^{\mathrm{L}}(t)|).$$

We still have to ensure that the definition of $T^{\mathrm{safe}}(t)$ is compatible with the requirement from Fact 3.1 regarding the time t_0 when the tour first returns to the origin. In case $|a^{\mathrm{L}}(t)| \geq |a^{\mathrm{R}}(t)|$, we get $t_0 \geq t_0^{\mathrm{L}}(t)$ with $t_0 = \rho|T^{\mathrm{greedy}}(t)| - 2|a^{\mathrm{R}}(t)| \geq (4\rho - 2)|a^{\mathrm{R}}(t)|$. Symmetrically, for $|a^{\mathrm{L}}(t)| < |a^{\mathrm{R}}(t)|$, we get $(4\rho - 2)|a^{\mathrm{L}}(t)| \geq t_0^{\mathrm{R}}(t)$. In either case, we can derive the following condition on ρ.

FACT 3.2. *The safe tour $T^{\mathrm{safe}}(t)$ fulfills inequality (3.1) in Fact 3.1 if and only if $2 \geq \rho \geq \frac{9+\sqrt{17}}{8}$.*

We are now ready to describe our algorithm (cf. Algorithm 1). We argued that it is a safe option to follow $T^{\mathrm{safe}}(t)$ in order to stay ρ-competitive, provided no further requests appear. It will turn out that it is indeed always good enough to follow $T^{\mathrm{safe}}(t)$, if possible. However, at time t, we may be too far from the extreme that $T^{\mathrm{safe}}(t)$ serves first in order to catch up with the safe tour, in which case we have to resort to secondary strategies. If at time t the server cannot reach $T^{\mathrm{safe}}(t)$, it instead bases its behavior on the greedy tour as an estimate for T^{OPT}. Surprisingly, this estimate turns out to be sufficient to obtain an optimal online algorithm. There are three situations that can occur if the safe tour cannot be reached at time t. If the online server is on the same side of the origin as the extreme that $T^{\mathrm{greedy}}(t)$ serves first, our algorithm decides to follow the greedy tour. If the online server is on the other side of the origin than the extreme that $T^{\mathrm{greedy}}(t)$ serves first, we have to ensure that the condition of Fact 3.1 is not violated. If the tour serving the nearer extreme first satisfies (3.1), the algorithm serves this extreme first, i.e., it serves the extremes in a different order than $T^{\mathrm{greedy}}(t)$. Otherwise, we can deduce from (3.1) being violated that we can afford to serve the opposite extreme first, i.e., to follow $T^{\mathrm{greedy}}(t)$.

Algorithm 1: UPDATE$(t, \sigma^{\mathrm{L}}(t), \sigma^{\mathrm{R}}(t), p_t)$ for the closed online TSP Problem

this function is called upon release of a new extreme request
Input: time t, unserved extreme requests $\sigma^{\mathrm{R}}(t)$ and $\sigma^{\mathrm{L}}(t)$, position p_t of the server
Output: closed Online TSP tour serving all unserved requests
$A \leftarrow \operatorname{argmax}_{g \in \{a^{\mathrm{R}}(t), a^{\mathrm{L}}(t)\}} |g|$; $a \leftarrow \operatorname{argmin}_{g \in \{a^{\mathrm{R}}(t), a^{\mathrm{L}}(t)\}} |g|$
if $T^{\mathrm{greedy}}(t) = T_{\mathrm{LR}}(t)$ **then**
  $(a_1(t); t_1) \leftarrow \sigma^{\mathrm{L}}(t)$; $(a_2(t); t_2) \leftarrow \sigma^{\mathrm{R}}(t)$
else
  $(a_1(t); t_1) \leftarrow \sigma^{\mathrm{R}}(t)$; $(a_2(t), t_2) \leftarrow \sigma^{\mathrm{L}}(t)$
end
if $t^{\mathrm{wait}} := \rho |T^{\mathrm{greedy}}(t)| - (|p_t - A| + |A| + 2|a|) \geq t$ **then** `// $T^{\mathrm{safe}}(t)$ can be reached`
(A) $T^{\mathrm{ALG}} \leftarrow \mathrm{waituntil}(t^{\mathrm{wait}}) \oplus \mathrm{move}(A) \oplus \mathrm{move}(a) \oplus \mathrm{move}(p_0)$
else if $\mathrm{sign}(p_t) = \mathrm{sign}(a_1(t))$ **or** $t + |p_t - a_2(t)| + |a_2(t)| < \frac{\rho |a_1(t)| - (2-\rho) t_1}{2\rho - 3}$ **then**
(B1,B2) $T^{\mathrm{ALG}} \leftarrow \mathrm{move}(a_1(t)) \oplus \mathrm{move}(a_2(t)) \oplus \mathrm{move}(p_0)$
else
(C) $T^{\mathrm{ALG}} \leftarrow \mathrm{move}(a_2(t)) \oplus \mathrm{move}(a_1(t)) \oplus \mathrm{move}(p_0)$
end
return T^{ALG}

THEOREM 3.1. *There is a* $(9 + \sqrt{17})/8 \approx 1.64$*-competitive algorithm for closed online* TSP *on the line.*

We obtain the main result of this section by analyzing each of the above cases.

4 Lower Bound for open online TSP

In this section, we consider open online TSP on the line and give a tight lower bound on the best-possible competitive ratio. Note that a lower bound of 2 is obvious: At time 1, we present a request either at -1 or 1, whichever is further away from the online server. The online tour has length at least 2 while the optimum tour has length 1. Remarkably, we are able to show a slightly larger bound that turns out to be tight.

THEOREM 4.1. *Let* $\rho \approx 2.04$ *be the second-largest root (out of the four real roots) of* $9\rho^4 - 18\rho^3 - 78\rho^2 + 210\rho - 107$*. There is no* $(\rho - \varepsilon)$*-competitive algorithm for open TSP on the line for any* $\varepsilon > 0$*.*

In the following, we fix any online algorithm ALG and $\rho' \in (2, \rho)$ and describe an adversarial strategy that forces $|T^{\mathrm{ALG}}|$ to be larger than $|T^{\mathrm{OPT}}|$ by a factor of at least ρ'. After the first request σ_0^{R}, which is to the right[2] of the origin, we alternatingly present leftmost and rightmost extreme requests, in the i-th iteration called σ_i^{L} and σ_i^{R}, respectively, depending on ALG's behavior. Roughly, a new leftmost request σ_i^{L} appears whenever the last rightmost request $\sigma_{i-1}^{\mathrm{R}}$ is served, and a new rightmost request σ_i^{R} appears when ALG has moved close enough to σ_i^{L}. Importantly, we will show that some pair $(\sigma_i^{\mathrm{L}}, \sigma_i^{\mathrm{R}})$ is *critical*, in the following sense.

DEFINITION 4.1. *We call the last two requests* $\sigma_0^\star = (a_0^\star; |a_0^\star|)$ *and* $\sigma_1^\star = (a_1^\star; |a_1^\star|)$ *of a request sequence with* $\mathrm{sign}(a_0^\star) \neq \mathrm{sign}(a_1^\star)$ *and* $0 < |a_0^\star| \leq |a_1^\star|$ critical *for* ALG *if the following conditions hold:*

(i) Both tours $\mathrm{move}(a_0^\star) \oplus \mathrm{move}(a_1^\star)$ *and* $\mathrm{move}(a_1^\star) \oplus \mathrm{move}(a_0^\star)$ *serve all the requests presented until time* $|a_1^\star|$*.*

(ii) ALG *serves both* $\sigma_0^\star$ *and* $\sigma_1^\star$ *after time* $|a_1^\star|$*, and* $p_{|a_1^\star|}$ *lies between* $a_0^\star$ *and* $a_1^\star$*.*

(iii) Let $k \in \{0, 1\}$ *be such that* ALG *serves* $\sigma_k^\star$ *before* $\sigma_{1-k}^\star$*. Then* ALG *serves* $\sigma_k^\star$ *no earlier than* $t^\star := (2\rho' - 2) \cdot |a_{1-k}^\star| + (\rho' - 2) \cdot |a_k^\star|$*.*

(iv) It holds that $|a_{1-k}^\star| / |a_k^\star| \leq 2$*.*

Indeed, we have the following lemma.

LEMMA 4.1. *If there is a request sequence with two critical requests for* ALG*, we can release additional requests such that* ALG *is not* $(\rho - \varepsilon)$*-competitive on the resulting instance.*

In the proof, we use the notation from Definition 4.1. We assume that $\mathrm{sign}(a_k^\star) \geq 0$; the other case is symmetric. For the sake of readability, we define $\sigma^{\mathrm{L}} = (a^{\mathrm{L}}; -a^{\mathrm{L}}) := \sigma_{1-k}^\star$ and $\sigma^{\mathrm{R}} = (a^{\mathrm{R}}; a^{\mathrm{R}}) := \sigma_k^\star$.

Conceptually, we want to present additional requests after $|a_1^*|$ so that ALG serves σ^{R} before σ^{L}. However, it will turn out that serving σ^{R} first is a

[2] We assume $p_1 \leq 0$; the other case is symmetrical.

mistake for ALG, compared with using the tour $T_{\mathrm{LR}} \coloneqq \mathrm{move}(\sigma^{\mathrm{L}}) \oplus \mathrm{move}(\sigma^{\mathrm{R}})$. Roughly, we make OPT follow the tour T_{LR} and then let it continue moving to the right until all requests are served by ALG. Accordingly, we will ensure that all additional requests we introduce coincide with OPT's position at their release time.

Assume that we could force ALG to serve σ^{L} immediately after σ^{R}, before serving any additional requests. In this case, we could simply introduce another request at a^{R} at time $|T_{\mathrm{LR}}|$, and, by Definition 4.1 (iii), we would have

$$\begin{aligned} |T^{\mathrm{ALG}}| &\geq t^\star + 2(|a^{\mathrm{R}}| + |a^{\mathrm{L}}|) \\ &= (2\rho' - 2) \cdot |a^{\mathrm{L}}| + (\rho' - 2) \cdot |a^{\mathrm{R}}| \\ &\quad + 2(|a^{\mathrm{R}}| + |a^{\mathrm{L}}|) \\ &= \rho'(2|a^{\mathrm{L}}| + |a^{\mathrm{R}}|) = \rho' |T^{\mathrm{OPT}}|, \end{aligned} \tag{4.2}$$

as claimed.

In general, however, ALG may not serve σ^{L} immediately after σ^{R}, for example by waiting for a while at a^{R} – which forces us to postpone the release of additional requests. Of course, ALG needs to start moving towards σ^{L} at some point to stay competitve if no new requests appear. Our goal is to balance these two effects by introducing one or two new requests.

The additional requests we use depend on the tour that ALG takes after time $|a_1^\star|$. Towards this, let $t^{\star\star}$ be the earliest possible time that a server starting in a^{R} at time $t^\star$ could serve σ^{L}, that is,

$$\begin{aligned} t^{\star\star} &\coloneqq t^\star + |a^{\mathrm{R}}| + |a^{\mathrm{L}}| \\ &= (2\rho' - 2) \cdot |a^{\mathrm{L}}| + (\rho' - 2) \cdot |a^{\mathrm{R}}| + |a^{\mathrm{R}}| + |a^{\mathrm{L}}| \\ &= (2\rho' - 1) \cdot |a^{\mathrm{L}}| + (\rho' - 1) \cdot |a^{\mathrm{R}}|. \end{aligned}$$

We characterize the trajectory of ALG at time $t \geq |a_1^\star|$ by the difference between $t^{\star\star}$ and the earliest possible time that ALG can still serve σ^{L}, if it aborts its tour at time t and takes the shortest tour serving σ^{R} (if needed) and then σ^{L}. Formally, for $t \geq |a_1^\star|$, we define

$$\mathrm{delay}(t) \coloneqq \begin{cases} t + 2|a^{\mathrm{R}}| + |a^{\mathrm{L}}| - p_t - t^{\star\star}, \\ \qquad \text{if } \sigma^{\mathrm{R}} \text{ not served at } t, \\ t + |a^{\mathrm{L}}| + p_t - t^{\star\star}, \\ \qquad \text{if } \sigma^{\mathrm{R}} \text{ served at } t \text{ but } \sigma^{\mathrm{L}} \text{ not}, \\ \text{undefined, else.} \end{cases}$$

It is easy to see that the following properties hold.

FACT 4.1. *Consider some t such that $\mathrm{delay}(t)$ is defined. Let $\mathcal{T}$ be the set of tours that start in position p_t at time t and, if* ALG *has not served σ^{R} at time t, that do not visit a^{L} before a^{R}. The following is true:*

(i) There is no tour $T \in \mathcal{T}$ that arrives at a^{L} earlier than $t^{\star\star} + \mathrm{delay}(t)$.
(ii) There is a tour $T \in \mathcal{T}$ that arrives at a^{L} at time $t^{\star\star} + \mathrm{delay}(t)$.

The following two lemmata state useful properties of the delay function that will be used to define the additional requests.

LEMMA 4.2. *There exists $W \geq 0$ with*

$$\mathrm{delay}\left(|T_{\mathrm{LR}}| + \frac{W}{\rho' - 1}\right) = W. \tag{4.3}$$

LEMMA 4.3. *With W as in Lemma 4.2,* ALG *serves σ^{R} no later than time $|T_{\mathrm{LR}}| + \frac{W}{\rho'-1}$.*

We will show that if we present an additional request at time $|T_{\mathrm{LR}}| + W/(\rho' - 1)$ (at a distance of $W/(\rho' - 1)$ to the right of σ^{R}) and ALG decides to serve σ^{L} before the new request, the ratio between ALG's and OPT's additional costs (Inequality (4.2)) is at least ρ'. If ALG can save time by serving the new request first and does so, we need to present yet another request.

It remains to show that we can define a request sequence (depending on ALG) that ends with a pair of critical requests. We use the following strategy:

- W.l.o.g. $p_1 \leq 0$. The first request is $\sigma_0^{\mathrm{R}} \coloneqq (1, 1)$.
- Whenever, at some time, called t_i^{L} in the following, a request at $\sigma_{i-1}^{\mathrm{R}}$ gets served, we present the new request $\sigma_i^{\mathrm{L}} \coloneqq (a_i^{\mathrm{L}} = -t_i^{\mathrm{L}}; t_i^{\mathrm{L}})$. Based on t_i^{L}, we define for $t \geq t_i^{\mathrm{L}}$ the two functions

  $$\ell_i^{\mathrm{L}}(t) \coloneqq (2\rho' - 3) \cdot t - (3 - \rho') \cdot t_i^{\mathrm{L}},$$
  $$\ell_i^{\mathrm{R}}(t) \coloneqq (4 - \rho') \cdot t - (2\rho' - 2) \cdot t_i^{\mathrm{L}},$$

  which can as well be viewed as lines in the path-time diagram.
- If at some time t_i^{R} after t_i^{L} ALG crosses $\ell_i^{\mathrm{L}}(t)$ or $\ell_i^{\mathrm{R}}(t)$, we present the request $\sigma_i^{\mathrm{R}} \coloneqq (a_i^{\mathrm{R}} = t_i^{\mathrm{R}}; t_i^{\mathrm{R}})$.
- We stop the procedure when one of the following cases occurs. (The pair $(\sigma_i^{\mathrm{L}}, \sigma_i^{\mathrm{R}})$ will be shown to be critical in these cases.)

  Case 1: ALG serves σ_i^{L} before σ_i^{R} if no new requests appear.

  Case 2: ALG serves σ_i^{R} not before time $(2\rho' - 2) \cdot t_i^{\mathrm{L}} + (\rho' - 2) \cdot t_i^{\mathrm{R}}$ if no new requests appear.

The intuition behind the lines ℓ_i^{L} and ℓ_i^{R} is the following: Suppose the position of ALG at t_i^{R} is on or to the right of ℓ_i^{L} and ALG decides to serve σ_i^{L}

before σ_i^{R} in case no new requests appear after t_i^{R}. Then the pair $(\sigma_i^{\mathrm{L}}, \sigma_i^{\mathrm{R}})$ satisfies Definition 4.1 (iii). The symmetric statement holds for ℓ_i^{R}. The following lemma ensures that, in each iteration, we obtain a (non necessarily critical) pair of unserved requests $(\sigma_i^{\mathrm{L}}, \sigma_i^{\mathrm{R}})$ and that Definition 4.1 (iv) is fulfilled.

LEMMA 4.4. *Let $i \geq 1$. At time t_i^{L},* ALG *is to the right of ℓ_i^{L} and ℓ_i^{R} and crosses one of them after t_i^{L} and before it serves σ_i^{L}. We also have that $t_i^{\mathrm{R}} \leq 2t_i^{\mathrm{L}}$.*

In the proof of Theorem 4.1, we show that Case 1 or 2 eventually occurs, and we formalize the above intuition to show along with Lemma 4.4 that the requests $(\sigma_i^{\mathrm{L}}, \sigma_i^{\mathrm{R}})$ are indeed critical.

5 Algorithm for open online TSP

In Algorithm 2, we describe an algorithm for the open online TSP on the line which achieves a competitive ratio of $\rho \approx 2.04$, matching the lower bound presented in Section 4.

THEOREM 5.1. *Algorithm 2 is ρ-competitive with $\rho \approx 2.04$ being the second-largest root of the polynomial $9\rho^4 - 18\rho^3 - 78\rho^2 + 210\rho - 107$.*

In the following, we discuss the different cases that can occur in Algorithm 2 and give an intuition of their interplay. Algorithm 2 is called every time a new extreme request is released. It then computes a new tour serving the current extreme requests and thus also all other requests between these extremes. It is important that we wait in certain cases to protect against the release of new extremes that force us to serve the extremes in a different order than we initially chose. Algorithm 2 lets the server return to the origin and wait there whenever possible, and moves to the extremes as late as possible. Intuitively, staying as close to the origin as possible has the benefit that the algorithm can delay the choice in which order to serve the extremes for as long as possible. We observe in the lower bound construction (Section 4), that a ρ-competitive algorithm may not serve a request too early, i.e., $|p_t|/t$ is bounded when a request is served. The exact bound on this ratio is computed in the proof of Lemma 4.4 and coincides with the bound in Lemma 5.1 for Algorithm 2.

LEMMA 5.1. *The position of the server in a tour computed by Algorithm 2 satisfies*

$$\frac{|p_t|}{t} \leq \frac{3\rho - 5}{-3\rho^2 + 9\rho - 4} \approx 0.58$$

for all times $t \geq 0$.

We decide the order in which we serve the extremes and possibly wait based on the current position p_t of the server, the current time t and the release times and positions of the single or the two extremes. The server always tries to move back to the origin and wait there as long as possible, i.e., it follows the tour $T_0 := \mathrm{move}(0) \oplus \mathrm{waituntil}(\infty)$ for as long as possible. We use the notation "T_0. until(condition(τ))" to denote the tour that follows T_0 until the first time t_0 that satisfies "condition(t_0)". Note that this may happen before the server reaches the origin.

We now discuss the different cases of the algorithm step-by-step. Let us first consider the simplest case where only one extreme request $\sigma_1 = (a_1; t_1)$ is present (cases (P1) and (E1) in Algorithm 2). The offline optimum OPT obviously cannot finish before time t_1 in this case. In order to guarantee ρ-competitiveness it is therefore sufficient to serve σ_1 at time ρt_1. Hence, we can afford to move to the origin and wait until the equation $t + |p_t - a_1| = \rho t_1$ is satisfied for the current time t and position p_t. This ensures that $|p_t|/t$ is bounded as stated in Lemma 5.1 (this is not clear and formally proved in the full version of the paper) and that we have the option to change our tour without much additional cost if an extreme on the other side is released later on. We call the resulting tour in this case, the *preferred tour* (case (P1)). It can happen, however, that we are only able to serve σ_1 later than time ρt_1, i.e. $t + |p_t - a_1| > \rho t_1$. This means that the until-condition is already satisfied and the server serves σ_1 immediately. We call this tour the *enforced tour* (case (E1)). The makespan of Algorithm 2 in this case is $|T^{\mathrm{ALG}}| = t_1 + |p_{t_1} - a_1|$. But we have $|T^{\mathrm{OPT}}| \geq t_1$ and also $|T^{\mathrm{OPT}}| \geq |p_t - a_1|$ as Algorithm 2 only visits points on the real line that must also be visited by OPT at some time. Overall, we have $|T^{\mathrm{ALG}}| \leq 2 \cdot \mathrm{OPT}$, implying ρ-competitiveness.

Now, consider the case where two extremes $\sigma_1 = (a_1; t_1)$ and $\sigma_2 = (a_2; t_2)$ are present. The two extremes define an interval

$$[a^{\mathrm{L}}(t), a^{\mathrm{R}}(t)] = [\min\{a_1, a_2\}, \max\{a_1, a_2\}].$$

Note that $a^{\mathrm{L}}(t) < p_t < a^{\mathrm{R}}(t)$ holds by definition of extreme requests.

If $0 \notin [a^{\mathrm{L}}(t), a^{\mathrm{R}}(t)]$ (case (O) in Algorithm 2), we let σ_1 denote the request closer to the origin, i.e., $|a_1| \leq |a_2|$. We then immediately serve σ_1 in order to ensure that Lemma 5.1 holds and $|p_t|/t$ stays small. We know that the offline optimum OPT cannot finish before time t_2. After serving σ_1 it is therefore safe to return to the origin and wait as long as we can to reach a_2 at time ρt_2. We can thus follow the tour T_0 after serving σ_1 at time $t + |p_t - a_2| \geq \rho t_2$. Possibly, this equation is already satisfied from the start and

Algorithm 2: For the open online TSP problem on the line.

This function is called every time a new extreme request is released.
Input: Current time t, current extremes, current position p_t
Output: The next part of the tour for the server
$T_0 := \text{move}(0) \oplus \text{waituntil}(\infty)$
if $\exists$ *only single extreme* $\sigma_1 = (a_1; t_1)$ **then**
(P1),(E1) **return** $T_0.\,\text{until}(\tau + |p_\tau - a_1| \geq \rho t_1) \oplus \text{move}(a_1)$
end
if $0 \notin [a^{\text{L}}(t), a^{\text{R}}(t)]$ **then**
$(\sigma_1 = (a_1; t_1),\ \sigma_2 = (a_2; t_2)) \leftarrow$ extremes such that $|a_1| \leq |a_2|$
(O) **return** $\text{move}(a_1) \oplus T_0.\,\text{until}(\tau + |p_\tau - a_2| \geq \rho t_2) \oplus \text{move}(a_2)$
else
$(\sigma_1 = (a_1; t_1),\ \sigma_2 = (a_2; t_2)) \leftarrow$ extremes such that $t_1 \leq t_2$
if $t + |p_t - a_1| \leq L^{\sigma_1,\sigma_2}$ **then**
(P2) **return** $T_0.\,\text{until}(\tau + |p_\tau - a_1| = L^{\sigma_1,\sigma_2}) \oplus \text{move}(a_1) \oplus \text{move}(a_2)$
else if $t + |p_t - a_2| \leq L^{\sigma_2,\sigma_1}$ **and** $|a_2| \leq \frac{3\rho-5}{(2\rho-2)(7-3\rho)}(\rho t_1 + (\rho-2)|a_1|)$ **then**
(A2) **return** $T_0.\,\text{until}(\tau + |p_\tau - a_2| = L^{\sigma_2,\sigma_1}) \oplus \text{move}(a_2) \oplus \text{move}(a_1)$
else
(E2) **return** $\text{move}(a_1) \oplus \text{move}(a_2)$
end
end

we serve σ_2 immediately.

Next, consider the case that $0 \in [a^{\text{L}}(t), a^{\text{R}}(t)]$ (cases (P2), (A2) and (E2)). Let now σ_1 be the request released first, i.e. $t_1 \leq t_2$, and $t = t_2$ be the current time. We have a lower bound of $|T^{\text{OPT}}| \geq t_1 + |a_1| + |a_2|$, irrespective of whether Opt serves σ_1 or σ_2 first. If we serve σ_1 first, which we call the *preferred tour* (case (P2)), we ensure that the tour produced by Algorithm 2 is not longer than $\rho|T^{\text{Opt}}|$ by satisfying the inequality

$$t + |p_t - a_1| + |a_1| + |a_2| \leq \rho(t_1 + |a_1| + |a_2|). \tag{5.4}$$

Now assume that Opt serves σ_2 first and a new request $\sigma_1' = (a_1; t_2 + |a_1| + |a_2|)$ appears at the same position as σ_1 when Opt arrives there. The new request does not increase the cost for Opt, which is still only lower bounded by $|T^{\text{OPT}}| \geq t_2 + |a_1| + |a_2|$. But our algorithm, which served σ_1 first, may be closer to σ_2 at the time when this new request appears and now has to go all the way back to the position of a_1 after serving σ_2. The intuition why it is sufficient to protect against this worst-case is that if σ_1' appears at a position further away from the origin, then this additional distance has to be traveled by Opt as well, and if σ_1' appears closer to the origin, it only benefits our algorithm. In order to ensure ρ-competitiveness in this scenario, the following inequality has to also be satisfied:

$$\begin{aligned} &t + |p_t - a_1| + 2(|a_1| + |a_2|) \\ &\quad \leq \rho(t_2 + |a_1| + |a_2|). \end{aligned} \tag{5.5}$$

If we now define

$$\begin{aligned} L^{\sigma_1,\sigma_2} := \min\{&\rho t_1 + (\rho-1)|a_1| + (\rho-1)|a_2|, \\ &\rho t_2 + (\rho-2)|a_1| + (\rho-2)|a_2|\}, \end{aligned}$$

then Inequalities (5.4) and (5.5) are simultaneously satisfied if and only if $t + |p_t - a_1| \leq L^{\sigma_1,\sigma_2}$. If this latter inequality is satisfied with some slack, we again follow the tour T_0 until it becomes tight, so that $|p_t|/t$ stays small and we are more flexible to change our tour when new requests appear.

In case the conditions for the preferred tour are not satisfied, we try to serve σ_2 first. This is called the *anticipated tour* (case (A2)). The inequalities that need to be satisfied in this case are the same as those for the preferred tour with σ_1 and σ_2 exchanged. Moreover, to ensure that $|p_t|/t$ is bounded as claimed in Lemma 5.1, the following inequality also needs to be satisfied if Algorithm 2 serves σ_2 first:

$$\begin{aligned} |a_2| &\leq \frac{3\rho - 5}{(2\rho-2)(7-3\rho)}\,(\rho t_1 + (\rho-2)|a_1|) \\ &\approx 1.21 \cdot t_1 + 0.02 \cdot |a_1|. \end{aligned} \tag{5.6}$$

Intuitively, the inequality ensures that σ_2 is not too far from the origin compared to σ_1, as otherwise we

would need to start too early to serve the extreme σ_2 and would violate the bound on $|p_t|/t$.

Lastly, in case neither the conditions for the preferred tour nor the anticipated tour are met, we immediately serve the earlier request σ_1 first and σ_2 directly afterwards. This is called the *enforced tour* (case (E2)). The main challenge in showing ρ-competitiveness in the analysis of the algorithm is to derive a better lower bound on Opt in this case by considering extremes that must have been released before.

6 Online Dial-A-Ride on the line

In this section we give a $\left(1+\sqrt{2}\right)$-competitive algorithm for (preemptive) open online Dial-A-Ride on the line. Let T_S^{Opt} denote the optimal tour over a set of requests S, starting at position $p_0 = 0$, and let R_t denote the set of released but not yet delivered requests at a time t. Our algorithm works as follows (cf. Algorithm 3): The server stays at position p_0 until the first request arrives. With every new arriving request, the server stops its current tour and returns to p_0, or stays at p_0 if it is already at the origin. The server starts following the tour $T_{R_t}^{\text{Opt}}$ at time $\sqrt{2} \cdot |T_{R_t}^{\text{Opt}}|$. By "unload" we denote the operation of unloading all requests the server is currently carrying at the current position. Formally, each such request $(a, b; t')$ is changed to $(p_t, b; t')$.

Algorithm 3: For the open online Dial-A-Ride problem with $c \geq 0$ fixed.

this function is called upon receiving a new request;
Input: new request σ, current position p_t, unserved requests R_t
Output: an open tour starting at p_t and serving all requests in R_t
return $\text{unload} \oplus \text{move}(p_0) \oplus \text{waituntil}(\sqrt{2} \cdot |T_{R_t}^{\text{Opt}}|) \oplus T_{R_t}^{\text{Opt}}$

We show that we get back to the origin in time whenever a new request is released.

Theorem 6.1. *Algorithm 3 is* $(1+\sqrt{2}) \approx 2.41$*-competitive for the preemptive open online* Dial-A-Ride *problem with capacity* $c \geq 1$.

Note 6.1. *Algorithm 3 can easily be modified to solve open* Dial-A-Ride *in general metric spaces and upholds the same competitive ratio. The proof remains the same.*

We also provide a lower bound for non-preemptive closed Dial-A-Ride on the line that improves the lower bound of 1.70 from [2].

Theorem 6.2. *No algorithm for the non-preemptive closed* Dial-A-Ride *problem on the line with fixed capacity* $c \geq 1$ *has competitive ratio lower than* $\rho = 1.75$.

7 The offline problem

Psaraftis et al. [21] show that open offline TSP on the line with release dates can be solved in quadratic time. For the closed variant they claim that the optimal tour has the structure [21, pp. 215–216]:

$$\text{waituntil}() \oplus \text{move}(A^{\text{R}}) \oplus \text{move}(A^{\text{L}}) \oplus \text{move}(p_0)$$

or

$$\text{waituntil}() \oplus \text{move}(A^{\text{L}}) \oplus \text{move}(A^{\text{R}}) \oplus \text{move}(p_0).$$

Here the waiting time at the origin is chosen maximally such that all requests are still served. We contradict this claim by showing that an optimal server tour may need to turn around arbitrarily many times.

Theorem 7.1. *For every* $k \in \mathbb{N}$*, there is an instance of closed* TSP *on the line such that any optimal solution turns around at least* $2k$ *times.*

Next, we show a dynamic program that solves closed offline TSP on the line in quadratic time. It is inspired by the one of Psaraftis et al. [21] for the open variant and a dynamic program of Tsitsiklis [22] for the same problem with deadlines instead of release times.

The algorithm (cf. Algorithm 4) relies on the fact that an optimal server tour has a 'zig-zag shape' with decreasing amplitude.

We index the requests by increasing positions. Then we compute for each index pair $i < j$ the completion time of two tours serving all requests to positions smaller than position a_i and all requests to positions larger than position a_j. We use $C_{i,j}^{+}$ for the best such tour ending at position a_j and $C_{i,j}^{-}$ for the best one ending at a_i. Starting with the largest difference $j - i$ and iteratively decreasing it, we recursively compute $C_{i,j}^{+}$ and $C_{i,j}^{-}$.

Theorem 7.2. *Algorithm 4 computes the minimum completion time of a server tour for offline TSP on the line in time* $O(n^2)$.

For the non-preemptive Dial-A-Ride problem on the line we show that the open and closed variant with release times are NP-hard. Without release times, we prove they are NP-hard for capacity $c \geq 2$. Our reductions are from the circular arc coloring problem, which is also used in a reduction for minimizing the sum of completion times of Dial-A-Ride on the line with capacity $c = 1$ [9].

Algorithm 4: Dynamic Program for Closed Offline TSP on the line.

Input: A set of requests $\sigma_i = (a_i; t_i)$ with $t_i \geq |a_i|$ for $-\ell \leq i \leq r$ and $a_i < a_{i+1}$ for $-\ell \leq i < r$.
Output: The minimum completion time $C_{\max}$ of a tour.
$C^+_{-\ell-1,r} \leftarrow t_r$.
$C^-_{-\ell,r+1} \leftarrow t_{-\ell}$.
for $i = -\ell, ..., r$ **do**
  $C^-_{i,r+1} \leftarrow \max\{t_i, C^-_{i-1,r+1} + a_i - a_{i-1}\}$.
end
for $j = r, .., \ell$ **do**
  $C^+_{-\ell-1,j} \leftarrow \max\{t_j, C^+_{\ell-1,j+1} + a_{j+1} - a_j\}$.
end
for $d = r + \ell, ..., 0$ **do**
  for $i = -\ell, .., r - d$ **do**
    $j \leftarrow i + d$.
    $C^-_{i,j} \leftarrow \max\{t_i, \min\{C^+_{i-1,j} + a_j - a_i, C^-_{i-1,j} + a_i - a_{i-1}\}\}$.
    $C^+_{i,j} \leftarrow \max\{t_j, \min\{C^+_{i,j+1} + a_{j+1} - a_j, C^-_{i,j+1} + a_j - a_i\}\}$.
  end
end
return $C_{\max} = C^+_{0,0}$.

THEOREM 7.3. *The non-preemptive open and closed offline* DIAL-A-RIDE *problem on the line are NP-complete. For capacity $c \geq 2$ this even holds when all release times are* 0.

In the classification of closed dial-a-ride problems in [9], offline TSP on the line is the problem $1|s = t, d_j|\text{line}|C_{\max}$. De Paepe et al. [9] claim Tsitsiklis [22] shows a polynomial algorithm, but this is for the open version. We solve the closed variant by giving a polynomial algorithm (Theorem 7.2) as well as a counterexample to the algorithm of Psaraftis et al. [21]. Our Theorem 7.3 shows the problem $1, \text{cap}1|d_j|\text{line}|C_{\max}$ in the same classification scheme is NP-hard. While this is implicitly claimed in [9], no proof is given. For the generalization to arbitrary capacity but without release dates, $1||C_{\max}$, Guan [13] showed hardness for capacity $c = 2$ and our new hardness proof handles any capacity $c \geq 2$.

References

[1] Foto Afrati, Stavros Cosmadakis, Christos H. Papadimitriou, George Papageorgiou, and Nadia Papakostantinou. The complexity of the travelling repairman problem. *Informatique Theorique et Applications*, 20(1):79–87, 1986.

[2] Norbert Ascheuer, Sven Oliver Krumke, and Jörg Rambau. Online Dial-a-Ride problems: Minimizing the completion time. *Proceedings of the 17th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 639–650, 2000.

[3] Mikhail J. Atallah and S. Rao Kosaraju. Efficient solutions to some transportation problems with applications to minimizing robot arm travel. *SIAM Journal on Computing*, 17(5):849–869, 1988.

[4] Giorgio Ausiello, Esteban Feuerstein, Stefano Leonardi, Leen Stougie, and Maurizio Talamo. Serving requests with on-line routing. In *Proceedings of the 4th Scandinavian Workshop on Algorithm Theory Aarhus on Algorithm Theory (SWAT)*, pages 37–48, 1994.

[5] Giorgio Ausiello, Esteban Feuerstein, Stefano Leonardi, Leen Stougie, and Maurizio Talamo. Competitive algorithms for the on-line traveling salesman. In *Proceedings of the 4th International Workshop on Algorithms and Data Structures (WADS)*, pages 206–217, 1995.

[6] Giorgio Ausiello, Esteban Feuerstein, Stefano Leonardi, Leen Stougie, and Maurizio Talamo. Algorithms for the on-line travelling salesman. *Algorithmica*, 29(4):560–581, 2001.

[7] Michiel Blom, Sven O. Krumke, Willem de Paepe, and Leen Stougie. The online TSP against fair adversaries. *INFORMS Journal on Computing*, 13(2):138–148, 2001.

[8] Moses Charikar and Balaji Raghavachari. The finite capacity dial-a-ride problem. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 458–467, 1998.

[9] Willem E. de Paepe, Jan Karel Lenstra, Jiri Sgall, René A. Sitters, and Leen Stougie. Computer-aided complexity classification of Dial-a-Ride problems. *INFORMS Journal on Computing*, 16(2):120–132, 2004.

[10] Esteban Feuerstein and Leen Stougie. On-line single-server Dial-a-Ride problems. *Theoretical Computer*

Science, 268(1):91–105, 2001.

[11] Greg N. Frederickson and Dih Jiun Guan. Nonpreemptive ensemble motion planning on a tree. *Journal of Algorithms*, 15(1):29–60, 1993.

[12] Paul C. Gilmore and Ralph E. Gomory. Sequencing a one state-variable machine: A solvable case of the traveling salesman problem. *Operations Research*, 12(5):655–679, 1964.

[13] Dih Jiun Guan. Routing a vehicle of capacity greater than one. *Discrete Applied Mathematics*, 81(1-3):41–57, 1998.

[14] Patrick Jaillet and Michael R. Wagner. Generalized online routing: New competitive ratios, resource augmentation, and asymptotic analyses. *Operations Research*, 56(3):745–757, 2008.

[15] Sven O. Krumke. Online optimization competitive analysis and beyond, 2001. Habilitation thesis.

[16] Sven O. Krumke, Willem E. de Paepe, Diana Poensgen, and Leen Stougie. News from the online traveling repairman. *Theoretical Computer Science*, 295(1-3):279–294, 2003.

[17] Sven O. Krumke, Luigi Laura, Maarten Lipmann, Alberto Marchetti-Spaccamela, Willem de Paepe, Diana Poensgen, and Leen Stougie. Non-abusiveness helps: An O(1)-competitive algorithm for minimizing the maximum flow time in the online traveling salesman problem. In *Proceedings of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX)*, pages 200–214, 2002.

[18] Eugene L. Lawler, Jan Karel Lenstra, Alexander H. G. Rinnooy Kan, and David B. Shmoys, editors. *The Traveling Salesman Problem; A Guided Tour of Combinatorial Optimization*. Wiley, Chichester, 1985.

[19] Maarten Lipmann. *On-Line Rounting*. PhD thesis, Technical University Eindhoven, 2003.

[20] Mark S. Manasse, Lyle A. McGeoch, and Daniel D. Sleator. Competitive algorithms for server problems. *Journal of Algorithms*, 11(2):208–230, 1990.

[21] Harilaos N. Psaraftis, Marius M. Solomon, Thomas L. Magnanti, and Tai-Up Kim. Routing and scheduling on a shoreline with release times. *Management Science*, 36(2):212–223, 1990.

[22] John N. Tsitsiklis. Special cases of traveling salesman and repairman problems with time windows. *Networks*, 22(3):263–282, 1992.